Object-Oriented Programming Language

04/30/2020

Lab Assignment 7

Exercise 1: **(about operator overloading)** Implement a class `VecFour` as a vector of four doubles. Implement suitable constructors and operators so the class can support the following client code:

```
int main ()
{
    VecFour a = VecFour(1.0,1.0,2.0,2.0) ;
    cout << "The vector \'a\' is: " << a << endl ;
    VecFour b ;
    cout << "Please input a vector: " ;
    cin >> b ;       // 0.0,1.0,2.0,5.0
    cout << "The vector you input is: " << b << endl ;
    VecFour c = 2.5*a ;
    cout << "The vector \'c\' is : " << c << endl ;
    c *= a;
    cout << "The vector \'c\' changes to : " << c << endl ;
    VecFour d ; // 0.0,0.0,0.0,0.0
    cout << "The vector \'d\' is : " << d << endl ;
    set<VecFour> coll{a, b, c, d};
    for (const auto& e : coll)
        cout << e << " ";
    cout << endl;
    return 0 ;
}
```

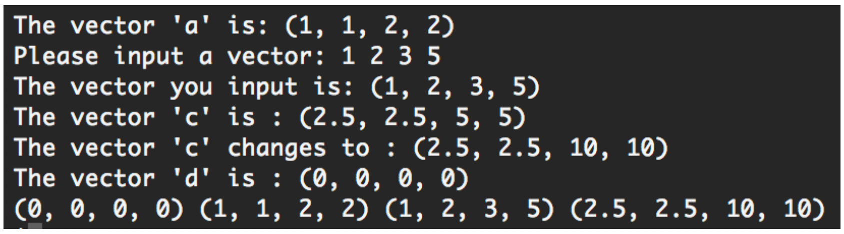
Please separate your code into a .cpp for the main, a .h for the `VecFour` class's declaration, and a .cpp for the `VecFour` class's definitions. Please declare and define the global functions in the .h with the `inline` keyword. The rules for multiplication follow standard inner product operation of a vector. That is:

$$(x_0 \quad x_1 \quad x_2 \quad x_3)\times(y_0 \quad y_1 \quad y_2 \quad y_3)=(x_0\times y_0 \quad x_1\times y_1 \quad x_2\times y_2 \quad x_3\times y_3)$$

$$(x_0 \quad x_1 \quad x_2 \quad x_3)\times\alpha=(x_0\times\alpha \quad x_1\times\alpha \quad x_2\times\alpha \quad x_3\times\alpha)$$

$$\alpha\times(x_0 \quad x_1 \quad x_2 \quad x_3)=(\alpha\times x_0 \quad \alpha\times x_1 \quad \alpha\times x_2 \quad \alpha\times x_3)$$

The rule for comparison follows standard distance operation of a vector. That is, the square root of the sum of components. A sample run looks like:



```
The vector 'a' is: (1, 1, 2, 2)
Please input a vector: 1 2 3 5
The vector you input is: (1, 2, 3, 5)
The vector 'c' is : (2.5, 2.5, 5, 5)
The vector 'c' changes to : (2.5, 2.5, 10, 10)
The vector 'd' is : (0, 0, 0, 0)
(0, 0, 0, 0) (1, 1, 2, 2) (1, 2, 3, 5) (2.5, 2.5, 10, 10)
```

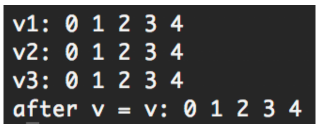
Exercise 2: **(about the rule of 3)** Implement a class `Vec` as an array of `int`s. The `n` is for the length and `v` is the pointer to the array of `int`s. The following are the data member parts of the class:

```
class Vec{
    int* v;
    int n;
};
```

The following is the test method which you CANNOT change:

```
int main(){
    Vec v;
    v.show("v1");

    {
        Vec v2(v);
        v2.show("v2");
    }

    {
        Vec v3;
        v3 = v;
        v3.show("v3");
    }

    v = v;

    v.show("after v = v");
}
```

Please add in a `show(…)` method according to the `main`, to match the following output:



```
v1: 0 1 2 3 4
v2: 0 1 2 3 4
v3: 0 1 2 3 4
after v = v: 0 1 2 3 4
```

The code seems to work fine, and life is great. But now, you are required to add in the destructor for the `Vec` class. The destructor has to delete the `v` pointer. You might need to add in suitable constructors and operators so the class can support the client code.